IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION
FOR:

APPLICATION CONTROL IN PEER-TO-PEER AD-HOC COMMUNICATION NETWORKS

INVENTORS:

Jan-Erik EKBERG, and
Pekka LAHTINEN

Morgan & Finnegan, L.L.P.
345 Park Avenue
New York, New York 10154-0053
(212) 758-4800
(212) 751-6849 (Facsimile)
www.MorganFinnegan.com

Attorneys for Applicant

44720 v1

APPLICATION CONTROL IN PEER-TO-PEER AD-HOC COMMUNICATION NETWORKS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application for letters patent is related to and incorporates by reference United States patent application serial number 10/284,135, titled "DEVICE DETECTION AND SERVICE DISCOVERY SYSTEM AND METHOD FOR A MOBILE AD HOC COMMUNICATIONS NETWORK", and filed in the United States Patent and Trademark Office on October 31, 2002. This application for letters patent is also related to and incorporates by reference United States continuation-in-part patent application serial number SS/XXX,YYY, titled "DEVICE DETECTION AND SERVICE DISCOVERY SYSTEM AND METHOD FOR A MOBILE AD HOC COMMUNICATIONS NETWORK", and filed in the United States Patent and Trademark Office on September 16, 2003. This application for letters patent is also related to and incorporates by reference United States patent application serial number SS/XXX,YYY, titled "MECHANISM FOR IMPROVING CONNECTION CONTROL IN PEER-TO-PEER AD-HOC NETWORKS", and filed in the United States Patent and Trademark Office on September 16, 2003. The assignee is the same in this patent application and the related patent applications.

FIELD OF THE INVENTION

[0002] The present invention relates, in general, to communication between devices connected to a wireless communication network. In particular, the present invention is a system and method for controlling access to an application program in a wireless device connected to a spontaneous and instant (ad-hoc) communications network.

BACKGROUND OF THE INVENTION

[0003] Short-range wireless systems have a range of less than one hundred meters, but may connect to the Internet to provide communication over longer distances. Short-range wireless systems include, but are not limited to, a wireless personal area network (PAN) and a wireless local area network (LAN). A wireless PAN uses low-cost, low-power wireless devices that have a typical range of ten meters. An example of a wireless PAN technology is the Bluetooth Standard. The Bluetooth Standard operates in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band and provides a peak air-link speed of one Mbps and a power consumption low enough for use in personal, portable electronics such as a personal digital assistance or mobile phone. A description of the Bluetooth communication protocol and device operation principles is in *Bluetooth Special Interest Group, Specification of the Bluetooth System*, version 1.1, volumes 1 and 2, February 22, 2001. Another example of a wireless PAN technology is a standard for transmitting data via infrared light waves developed by the Infrared Data Association (IrDA), a group of device manufacturers. IrDA ports enable computers, such as a laptop, or devices, such as a printer, to transfer data from one device to another without any cables. IrDA ports support roughly the same transmission rates as traditional parallel ports and the only restriction on their use is that the two devices must be within a few feet of each other and have a clear line of sight. A wireless LAN is more costly than a wireless PAN, but has a longer range. An example of a wireless LAN technology is the IEEE 802.11 Wireless LAN Standard and the HIPERLAN Standard. The HIPERLAN Standard operates in the 5 GHz

Unlicensed-National Information Infrastructure (U-NII) band and provides a peak air-link speed between ten and one hundred Mbps.

[0004] An ad-hoc network is a short-range wireless system comprising an arbitrary collection of wireless devices that are physically close enough to exchange information. An ad-hoc network is constructed quickly with wireless devices joining and leaving the network as they enter and leave the proximity of the remaining wireless devices. An ad-hoc network also may include one or more access points, that is, stationary wireless devices operating as a stand-alone server or as gateway connections to other networks.

[0005] In the future, the Bluetooth Standard will likely support the interconnection of multiple piconets to form a multi-hop ad-hoc network, or scatternet. In a scatternet, a connecting device forwards traffic between different piconets. The connecting device may serve as a master device in one piconet, but as a slave device or a master device in another piconet. Thus, the connecting devices join the piconets that comprise a scatternet by adapting the timing and hop sequence to the respective piconet and possibly changing the roles that they serve from a master device to a slave device.

[0006] A Bluetooth device includes, but is not limited to, a mobile telephone, personal or laptop computer, radio-frequency identification tag, and personal electronic device such as a personal digital assistant (PDA), pager, or portable-computing device. Each Bluetooth device includes application and operating system programs designed to find other Bluetooth devices as they enter and leave the communication range of the network. The requesting Bluetooth device in a client role and the responding Bluetooth device in a server role establish a proximity link between the two devices. The requesting and responding Bluetooth device use the proximity link and a service discovery protocol to discover the services offered by the other Bluetooth device and how to connect to those services.

[0007] In a traditional computing environment, an application program that is running in a computer is resident in the memory of the computer and is constrained by factors such as the memory size, processor speed, and resources. Typically, these factors do not impose limits on the application. The user controls the application (i.e., when an application is started, closed, and its relationship to other applications) using a shell program or a graphical desktop environment. The ability to auto-start or pre-configure the application exists, but only in the context of Plug-n-Play drivers and interfaces.

[0008] In a wireless computing environment, the computing environment necessitates strict application control in terminal devices. First, the number of bytes of memory that the user interface requires in a mobile device restricts the ability to run applications in parallel. Second, a user typically cannot control the establishment of a proximity connection between two peer devices. The user may know that there is a high probability of establishing the proximity connection, but cannot reliably predict the time or place of the establishment. Third, when multiple applications must be presented, the order that the applications will be presented to a user depends on factors such as the user's preferences and the configuration of the server. The server may combine several applications and run those applications in a certain order because the server's instructions indicate that the certain order will optimize the experience for the user. For example, a browsing application will run first to view a movie file, then a banking application will run to purchase a ticket, followed by a ticketing application to accept the purchased ticket, and finally, a convenience application will run to change the telephone ring to silent mode. Fourth, an application can be dynamically loaded and run on a terminal device (e.g., applets). In addition to security issues, this ability of the terminal device raises issues regarding the user's control of the terminal device.

[0009] Thus, there is a need for a system and method for controlling access to an application program in a wireless device connected to a spontaneous and instant (ad-hoc) communications network. The system and method will allow a user or service provider to create a rule set that describes the desired behavior of the application programs. The rule set will define the automatic launching of the application programs and the allowed behavior of the application programs following the establishment of a proximity connection. The present invention addresses this need.

SUMMARY OF THE INVENTION

[0010] A computer system, method, and computer program product for controlling access to an application program in a wireless device connected to an ad-hoc communications network. The method comprises sending an inquiry message to the ad-hoc communications network, receiving a response to the inquiry message from a nearby wireless device, choosing a selected application from a list of application programs, and examining control parameters associated with the selected application. In one embodiment, the control parameters dictate a behavior of the selected application such as allowing communication with the selected application, refusing communication with the selected application, downloading the selected application, or distributing the selected application. When a nearby wireless device includes a matching application, the method further comprises sending a connection request to the nearby wireless device, receiving an accept connections message from the nearby wireless device, launching the selected application, and sending a service request to connect the selected application and the matching application. When a user closes the selected application that was launched, the method further comprises erasing the selected application. In one embodiment, to choose the selected application, the method further comprises retrieving an entry from an application directory stored in a middleware layer. Alternatively, the choice of the selected application is based on a priority assigned to the entry.

[0011] In another embodiment, the method comprises receiving an inquiry message, sending a response to the inquiry message, receiving a connection request, sending an accept connections message, receiving a service request to connect to an application, and examining control parameters associated with a matching application program for the application. In one embodiment, the control parameters dictate a behavior of the selected application such as allowing communication with the selected application, refusing communication with the selected application, downloading the selected application, or distributing the selected application. In another embodiment, the method further comprises launching the matching application, and receiving a service request to connect the selected application and the matching application. When a user closes the selected application that was launched, the method further comprises erasing the selected application.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The accompanying figures best illustrate the details of the system and method for launching and controlling application programs resident in wireless devices in a spontaneous and instant (ad-hoc) communications network, both as to its structure and operation. Like reference numbers and designations in these figures refer to like elements.

[0013] Figure **1** is a network diagram that illustrates the interaction of the devices that comprise a mobile ad-hoc communications network, in accordance with one embodiment of the present invention.

[0014] Figure **2A** is a block diagram that illustrates the hardware and software components comprising server **110** shown in Figure **1**, in accordance with one embodiment of the present invention.

[0015] Figure **2B** is a block diagram that illustrates the hardware and software components comprising terminal **120** shown in Figure **1**, in accordance with one embodiment of the present invention.

[0016] Figures **3A–3B** are flow diagrams of an embodiment of a process that accesses the control parameters stored in the distributed application directory.

[0017] Figures **4A–4D** are flow diagrams of various embodiments of a process for application program control in a mobile ad-hoc communications network.

[0018] Figure **5** is a flow diagram of an embodiment of a process that illustrates the message flow during establishment of a communication session between terminal X and terminal Y in a mobile ad-hoc communications network.

DETAILED DESCRIPTION OF THE INVENTION

[0019] Figure **1** is a network diagram that illustrates the interaction of the devices that comprise a mobile ad-hoc communications network, in accordance with one embodiment of the present invention. In one embodiment, the mobile ad-hoc communications network is a Bluetooth piconet that includes one master device and up to seven active slave devices. As shown in Figure **1**, piconet **100** includes server **110** and five instances of terminal **120**. Server **110** maintains the network clock and is the communication manager for each instance of terminal **120**. Server **110** typically initiates an exchange of data with an instance of terminal **120**. Two instances of terminal **120** typically communicate through the server **110** however, if two instances of terminal **120** communicate directly, one instance will assume the role of server, or master, and the other instance will assume the role of client, or slave.

[0020] Each device in the mobile ad-hoc communications network will either assume the role of a terminal device or a server device. A terminal device is a consumer of services that a single user operates. A terminal device includes devices such as a mobile phone or PDA. A server is typically a stationary device and only produces services. A server device creates a hotspot around them for using their services. "Hotspot" refers to the radio coverage area provided by the server device for detecting devices and discovering services offered by the applications hosted in the server. If the server device is not stationary, one of the terminal devices in the network will assume the role of application directory server and perform device detection and service discovery functions for the remaining terminal devices in the network. The disclosed invention introduces two roles among such terminal devices, application directory servers and terminals, where application directory servers serve terminals in device detection and service discovery. If stationary servers with hotspots exist, servers typically act as application directory servers however, device detection and service discovery is possible without such a stationary server because one of the terminals will assume the application directory server duties.

[0021] The disclosed invention assigns an identifier to each application placed under control. In one embodiment, the identifier is a non-unique identifier that abstractly identifies the application. In another embodiment, the identifier specifies a function that the application performs. In another embodiment, the identifier specifies a communication protocol that the application uses to communicate. Thus, the identifier may indicate that several occurrences of an application each occurrence authored in a different computer language, or targeted to run on a different hardware platform or fulfill a different application role may be considered to be the same because they can interoperate and fulfill the same function. However, in yet another embodiment, the identifier is a unique identifier that identifies the application.

[0022] Figure **2A** is a block diagram that illustrates the hardware and software components comprising server **110** shown in Figure **1**, in accordance with one embodiment of the present invention. Server **110** is a general-purpose wireless device. Bus **200** is a communication medium that connects keypad **201**, display **202**, central processing unit (CPU) **203**, and radio frequency (RF) adapter **204** to memory **210**. RF adapter **204** connects via a wireless link to terminal **120** and is the mechanism that facilitates network traffic between server **110** and terminal **120**.

[0023] CPU **203** performs the methods of the disclosed invention by executing the sequences of operational instructions that comprise each computer program resident in, or operative on, memory **210**. Memory **210** includes operating system software **211**, application programs **212**, and middleware software **220**. Operating system software **211** controls keypad **201**, display **202**, RF adapter **204**, and the management of memory **210**. Application programs **212** control the interactions between a user and server **110**. Middleware software **220** includes an application program interface (API) **221** that help an application program running on server **110** find and communicate with a counterpart application running on terminal **120**. To quickly locate each application, middleware software **220** also includes application directory **230** to track, for each application that is resident in each device in piconet **100**, a reference to the device storing the application, an identifier for the application, the role that the application performs, and the control parameters that define the user or service provider rules for controlling the application. In one embodiment, the reference to the device storing the application is the MAC address of the device.

[0024] Figure **2B** is a block diagram that illustrates the hardware and software components comprising terminal **120** shown in Figure **1**, in accordance with one embodiment of the present invention. Terminal **120** is a general-purpose wireless device. Bus **250** is a communication medium that connects keypad **251**, display **252**, CPU **253**, and RF adapter **254** to memory **260**. RF adapter **254** connects via a wireless link to server **110** or another terminal **120** and is the mechanism that facilitates network traffic between server **110** and terminal **120**.

[0025] CPU **253** performs the methods of the disclosed invention by executing the sequences of operational instructions that comprise each computer program resident in, or operative on, memory **260**. Memory **260** includes operating system software **261**, application programs **262**, and middleware software **270**. Operating system software **261** controls keypad **251**, display **252**, RF adapter **254**, and the management of memory **260**. Application programs **262** control the interactions between a user and terminal **120**. Middleware software **270** includes an API **271** that help an application program running on terminal **120** find and communicate with a counterpart application running on server **110** or another terminal **120**. To quickly locate each application, middleware software **270** also includes application directory **280** to track, for each application that is resident in each device in piconet **100**, a reference to the device storing the application, an identifier for the application, the role that the application performs, and the control parameters that define the user or service provider rules for controlling the application. In one embodiment, the reference to the device storing the application is the MAC address of the device.

[0026] In one embodiment, the configuration of memory **210** and memory **260** is identical. In another embodiment, the configuration of memory **210** and memory **260** only includes the software necessary to perform the essential tasks of server **110** and terminal **120**, respectively. For example, if terminal **120** needs to receive a general inquiry access code, but does not need to send a general inquiry access code message, only the software that receives this message will reside in memory **260**.

[0027] In the disclosed invention, the distributed application directory stored in the middleware software is a database that makes it possible for a device to know something of the requirements and wishes of peer devices to which it connects. The database also contains information of local applications and their requirements. The information includes control parameters, or combinations of control parameters, as well as priority information, indicating importance of the application set by the user. These control parameters are stored in the distributed application directory, or database, and are enforced by the middleware software. In one embodiment, there are three categories of control parameters, application states, user-defined application settings, and macros (i.e., combinations of user-defined application settings).

Application States

[0028] "Installed" – An application program state indicating that the application program is resident and installed in the local memory of a wireless device. When an application program is installed, the local memory includes a binary, or digital, image of the application program.

[0029] "In-Machine" – An application program state indicating that the application program is available to the middleware software, but a binary, or digital, image of the application program is not installed into the local memory. The In-Machine state is the result of an auto-launchable, non-persistent distribution of an application program.

[0030] "Running" – An application program state indicating that the application program is currently running in the wireless device. An application program that is auto-launched will achieve the Running state when it launches and is able to communicate with peer devices. However, even though an application program may not be marked as an Auto-Launch application program, the application program can still be started manually by a user and reach the Running state by user interaction.

User-Defined Application Settings

[0031] Auto-Launch – A user may configure an application program to automatically start in a wireless device when a possible communication opportunity is available. The Auto-Launch setting is available for an application program regardless of the application state, Installed, In-Machine, or Running.

[0032] Auto-Launch Priority – A numerical value in a given range (e.g., the range 0–127) that defines the willingness for a user to automatically launch an application program when several applications need to be automatically launched using a connection between pairs of wireless devices. The first application program to be launched is the application program having the highest sum of the two Auto-Launch Priority values, the local application priority and the application priority to the peer device. An application program already running in either device takes absolute priority over an application program that has not yet been launched. The example that follows illustrates this user-defined application setting.

[0033] Refused – A wireless device that is receiving connection requests may mark an application program as being banned from running on the wireless device. If this value is set, the associated application program will never be launched, moved, or proposed for download to the wireless device.

[0034] Wanted – An application program that is not installed in a wireless device, but that a user wants to run is a Wanted application program. This setting is an authorization by the user to download and install a binary image of the application program from another wireless device.

[0035] Distributable – An application program setting denoting that a peer device, typically in a server role, is prepared to distribute binary images of the associated application program to connected peer devices.

[0036] Erase-After-Use – A peer device, typically in a server role, can configure an application program to be Distributable and non-persistent. Thus, the application program can be given to a peer device in order to establish communication, but the application program will never be installed on the peer device because it is automatically erased from the peer device after the use. Typical examples of this type of application program includes banking clients or multi-player game clients.

Macros

[0037] Auto-Download – If the middleware software database information indicates that an application program is not installed in a local device, is an Auto-Launch or Wanted application, has been marked by a peer device as Distributable, as well as Auto-Launchable or

Running, the application program should be downloaded and eventually launched. However, if the non-persistent flag is set, the application program may disappear after being used.

[0038] Downloadable – If the middleware software database information indicates that an application program is marked as Distributable in the peer, and is locally not an Auto-Launch application or a Wanted application, and is neither Installed nor Refused, then the application program is available to be downloaded and installed. A common case satisfying the above requirement is that there is no mention about the application in the local device, and it is marked as Distributable in the peer device.

[0039] Auto-Launch-Everything – This macro has certain limits, but is accomplished by having an Auto-Launch application program in a client wireless device that takes information of Distributable application programs in a server wireless device, and configures the middleware software to Auto-Launch those application programs. However, the user may define and introduce some restrictions.

[0040] Transfer and State Indications – When a device automatically launches an application, the device changes the state of the application from IDLE to RUNNING. Similarly, when a device terminates a running application, the device changes the state of the application from RUNNING to IDLE. This dynamic information is exchanged as part of the other application settings, because it is necessary, for example, when calculating the priority of the application. In addition, two application parameters, CLOSE and RELEASE-HISTORY, are never part of the parameter set and are only added and removed as additional data when parameter exchanges take place between specific peer devices. CLOSE is an indication to a given peer that the application session between the local device and the peer is closed, although the local application continues in a RUNNING state. CLOSE is used by server applications with respect to their clients. Normally, the fact that an application has been run and terminated between two peers is stored, and inhibits re-triggering of the same application as long as the data connection exists between the two devices. In some cases we want an application to be run again (e.g., the user starts the application manually after it has been automatically run once). In this case the RELEASE-HISTORY is sent to the peer device, releasing the memory regarding the given application, and a new session for the application in question can be established.

[0041] Figure **3A** and Figure **3B** are flow diagrams of an embodiment of a process that accesses the control parameters stored in the distributed application directory. Figure **3A** illustrates a portion of the process that accesses the control parameters to launch an application and enable two devices to communicate via the application. Furthermore, Figure **3A** corresponds to the example rule set macro that follows as "Example 1 – Macro". Figure **3B** illustrates another portion of the process that accesses the control parameters to determine whether an application is runnable. Furthermore, Figure **3B** corresponds to the example rule set macro that follows as "Example 2 – Macro Ordering".

[0042] The portion of the process shown in Figure **3A** illustrates the logic for starting application A when it is runnable in device D and peer P (step **300**). The process first determines whether application A is in device D (step **301**). If application A is not in device D, the process copies application A from peer P to device D (step **302**) and installs application A either permanently or temporarily (step **303**). If application A is in device D, the process then determines whether application A is in peer P (step **304**). If application A is not in peer P, the process copies application A from device D to peer P (step **305**). If application A is in peer P, the process then determines whether application A is running in peer P and device D (step **306**). If application A is running in peer P and device D, the process establishes a link connection between device D and peer P and allows application A to communication until termination (step **307**). If application A is not running in peer P and device D, the process then determines whether application A is running in device D (step **308**). If application A is not running in device D, the process starts application A in device D (step **309**). If application A is running in device D, the process then returns to the beginning of the logic for starting application A (step **300**).

[0043] The portion of the process shown in Figure **3B** illustrates the logic for determining whether application A is runnable in device D which is connected to peer P (step **310**). The process first determines whether application A has been run before during the link connection between device D and peer P (step **311**). If application A has been run before during the link connection between device D and peer P, application is not runnable. If application A has not been run before during the link connection between device D and peer P, the process then determines whether the control parameters for application A in device D or peer P indicate that application A is refused (step **312**). If application A is refused, application A is not runnable. If application A is not refused, the process then determines whether application A is in device D (step **313**). If application A is in device D, the process then determines whether application A is in peer P (step **314**). If application A is in peer P, the process then determines whether application A is running or auto-launchable in device D and peer P (step **315**). If application A is running or auto-launchable in device D and peer P, application A is runnable. If application A is not running or auto-launchable in device D and peer P, application A is not runnable. If application A is not in peer P, the process then determines whether application A is distributable in device D (step **316**). If application A is distributable in device D, the process then determines whether application A is running or auto-launchable in device D and peer P (step **315**). If application A is running or auto-launchable in device D and peer P, application A is runnable. If application A is not running or auto-launchable in device D and peer P, application A is not runnable. If application A is not distributable in device D or if application A is not in device D, the process then determines whether application A is in peer P (step **317**). If application A is in peer P, the process then determines whether application A is distributable in peer P (step **318**). If application A is distributable in peer P, the process then determines whether application A is running or auto-launchable in device D and peer P (step **315**). If application A is running or auto-launchable in device D and peer P, application A is runnable. If application A is not running or auto-launchable in device D and peer P, application A is not runnable. If application A is not in peer P, application A is not runnable.

[0044] Example 1 that follows is an example of a rule set macro for installing and starting application programs. In this example, L indicates a local wireless device, P indicates a peer device, COMM indicates that communication is possible, PULL or PUSH indicates the transfer of the installation package, LAUNCH indicates the launching of the application program, and INSTALL indicates the installing of an application program. This example omits all references to version and role control.

```
RUNNABLE_X = AUTOLAUNCH_X AND (INSTALLED_X OR IN_MACHINE_X) AND
NOT REFUSED_X

PULL_L = NOT INSTALLED_L AND NOT IN_MACHINE_L AND NOT REFUSED_L
AND DISTRIBUTABLE_P AND (WANTED_L OR AUTOLAUNCH_L)

PUSH_L = DISTRIBUTABLE_L AND NOT INSTALLED_P AND NOT
IN_MACHINE_P AND NOT REFUSED_P AND (WANTED_P OR AUTOLAUNCH_P)

DOWNLOADABLE_L = NOT INSTALLED_L AND NOT IN_MACHINE_L AND NOT
REFUSED_L AND DISTRIBUTABLE_P AND (NOT WANTED_L AND NOT
AUTOLAUNCH_L)
```

Example 1 – Macro

[0045] A state machine evaluates the macro shown in Example 1. The macro is evaluated for each application program separately with the additional dimensions of version control and roles. If ordering is necessary because many operations cannot be performed in parallel, the macro defines the ordering as well. The first priority is assigned to communicate, and the last priority is assigned to start the download of the application program. Example 2 illustrates one embodiment of the macro defining the ordering.

```
IF (RUNNING_L AND RUNNING_P) THEN COMM

ELSE IF (NOT RUNNING_L AND RUNNABLE_L AND (RUNNING_P OR
RUNNABLE_P)) THEN LAUNCH

ELSE IF PULL_L AND NOT PULLING THEN PULL

ELSE IF PUSH_L AND NOT PUSHING THEN PUSH

ELSE IF NOT INSTALLED_L AND IN_MACHINE_L AND NOT
NON_PERSISTENT_L AND NOT NON_PERSISTENT_P THEN INSTALL

ELSE IF DOWNLOADABLE_L THEN START DOWNLOAD APPLICATION
```

Example 2 – Macro Ordering

[0046] In one embodiment, the launching order for applications can also be set by selective database distribution in a strictly client-server architecture. However, an ordering can also be achieved by not setting applications to be Auto-Launch application programs and by dynamically altering the running flag.

[0047] In another embodiment, the Auto-Launch priority is a useful tool, but device-dependent. It works in a peer-to-peer or local setting when a client is talking to a server, but in a network there are possible dead-lock situations. This is primarily because the server device probably serves all customers and applications simultaneously, and only uses the priority to try to affect the launching order in client devices. Thus, the application programs should be prepared to close themselves if a peer device is not located.

Auto-Launch Priority Example

[0048] The Auto-Launch Priority order is best described by the following example. Two peer devices, Device A and Device B establish a connection. Device A and Device B have the following applications Installed and marked as Auto-Launch application programs.

Device A:	CRASH-GAME (priority 100) SHARE-A-JOKE (priority 52) COPY-HOMEWORK (priority 36)
Device B:	SHARE-A-JOKE (priority 43) COPY-HOMEWORK (priority 71)

[0049] In this example, Device A and Device B have two Auto-Launch application programs in common, COPY-HOMEWORK and SHARE-A-JOKE. COPY-HOMEWORK has an Auto-Launch Priority of 36 + 71 = 107. SHARE-A-JOKE has an Auto-Launch Priority of 52 + 43 = 95. Thus, COPY-HOMEWORK will start first because its Auto-Launch Priority of 107 is greater than the Auto-Launch Priority of 95 for SHARE-A-JOKE.

[0050] A necessary extension to this rule (to minimize deadlocks) is that application programs that are running in the peer device take absolute priority over application programs that are merely Auto-Launch programs. Thus, when Device A and Device B instead have the following applications Installed and marked as Auto-Launch application programs, the Auto-Launch Priority order should result in the order SHARE-A-JOKE, CRASH-GAME, COPY-HOMEWORK.

Device A:	CRASH-GAME (priority 100) SHARE-A-JOKE (priority 52) COPY-HOMEWORK (priority 36)
Device B:	SHARE-A-JOKE (priority 43) (running) COPY-HOMEWORK (priority 71) CRASH-GAME (priority 0)

[0051] Figures **4A–4D** are flow diagrams of various embodiments of a process for application program control in a mobile ad-hoc communications network. The ad-hoc communications network connects a number of devices. Figures **4A–4D** illustrate two of these devices, source device **400**, and peer device **450**.

[0052] Source device **400** initiates the process shown in Figure **4A** by sending an inquiry request to the ad-hoc communications network (step **401**). Peer device **450**, one of the devices in the ad-hoc communications network that is in inquiry scan mode, receives the inquiry request (step **451**) and responds by sending an inquiry response message (step **452**). In one embodiment, the inquiry response message is a Bluetooth inquiry result command modified to indicate that peer device **450** includes a middleware layer. Source device **400** receives the inquiry response message (step **402**). Source device **400** accesses the local combined application directory (step **403**) stored in the middleware software portion of the memory for source device **400**. Source device **400** chooses a selected application from the local combined application directory (step **404**) and examines the control parameters associated with the selected application (step **405**).

[0053] The control parameters provide several user-defined alternatives for each application, as well as, the ability to combine alternatives into a macro. The process first determines whether the control parameters allow a connection between the selected application running on source device **400** and a matching application running on peer device **450** (step **406**). If the control parameters refuse a connection for this application on source device **400**, the process exits. If the control parameters allow a connection for this application on source device **400**, the process then determines whether the application program is resident in source device **400** (step **407**).

[0054] If the application program is resident in source device **400**, source device **400** sends a paging request message (step **408**). Peer device **450** receives the paging request message (step **453**) and sends a paging accept message in response (step **454**). Source device **400** receives the paging accept message (step **409**) and launches the selected application (step **410**). Once the selected application is running, source device **400** sends a service discovery request (step **411**). Peer device **450** receives the service discovery request (step **455**) and decides whether to refuse the connection (step **456**) based on the control parameters for the matching application for the selected application. If the control parameters specify to refuse the connection to the matching application, peer device **450** exits. If the control parameters specify to allow the connection to the matching application, peer device **450** sends a response to the service discovery request (step **457**). Source device **400** receives the response (step **412**) and communication begins between the selected application and the matching application. When the communication stops, source device **400** and peer device **450** may erase the selected application and the matching application, respectively, if the control parameters specify to erase after use.

[0055] If the application program is not resident in source device **400**, source device **400** then determines whether the application program is distributable (step **413**). If the application program is not distributable, the process exits. If the application program is distributable, source device **400** sends an offer for the selected application (step **414**). Peer device **450** receives the offer for the selected application (step **458**) and sends a request for the selected application in response (step **459**). Source device receives the request for the selected application (step **415**) and sends the selected application in response (step **416**). Peer device **450** receives the selected application (step **460**). Source device **400** then determines whether the application program is downloadable (step **417**). If the application program is not downloadable, the process exits. If the application program is downloadable, source device **400** sends a request for the selected application (step **418**). Peer device **450** receives the request for the selected application (step **461**) and sends the selected application in response (step **463**). Source device receives the download of the selected application (step **419**). To start the newly downloaded application program, the process repeats by determining whether the control parameters allow a connection for the newly downloaded application on source device **400** (step **406**).

[0056] Figure **4C** illustrates a process for determining a preferred application from the applications found in peer device **450** after connection establishment between source device **400** and peer device **450**. The process shown in Figure **4C** begins by prioritizing the applications in source device **400** (step **420**). Typically, the user of source device **400** accesses the graphical user interface to prioritize the applications. In one embodiment, the prioritization includes specifying a preferred application from the applications that source device **400** can access. In another embodiment, the prioritization includes ordering from most important to least important every application that source device **400** can access. In yet another embodiment, the prioritization includes ordering from most important to least important a portion of the applications that source device **400** can access. Once the applications are prioritized, source device **400** sends an inquiry request to the ad-hoc communications network (step **421**). Peer device **450**, one of the devices in the ad-hoc communications network that is in inquiry scan mode, receives the inquiry request (step **463**) and responds by sending an inquiry response message (step **464**). In one embodiment, the inquiry response message is a Bluetooth inquiry result command modified to indicate that peer device **450** includes a middleware layer. Source device **400** receives the inquiry response message (step **422**). Source device **400** examines the inquiry response message to determine whether the inquiry response message includes an indication that peer device **450** may include the middleware layer (step **423**). If the inquiry response message does not include the indication, the process exits. If the inquiry response message includes the indication, source device **400** conducts paging and service discovery with peer device **450** to establish a link connection (step **424** and step **465**). Following establishment of the link connection, source device **400** confirms whether peer device **450** includes the middleware layer (step **425**). In one embodiment, a recognition request message and subsequent response message will confirm whether peer device **450** includes the middleware layer. If peer device **450** does not include the middleware layer, the process exits. If peer device **450** includes the middleware layer, source device **400** accesses the combined directory (step **426**) and examines the control parameters associated with the prioritized applications and selects the preferred application for launching (step **427**). Subsequently, peer device **450** launches the requested application (step **466**).

[0057] In one embodiment, the distributed directory information includes the control parameters associated with the prioritized applications (i.e., preference information). Thus, the preference information is included in the distribution of the application directory for the peer device and determining whether to launch an application is a decision made by the source device locally using locally stored control parameters.

[0058] Figure **4D** illustrates a process for selecting peer device **450**, before connection establishment, from a number of nearby devices because peer device **450** includes at least one preferred application. The process shown in Figure **4D** begins by prioritizing the applications in source device **400** (step **428**). Typically, the user of source device **400** accesses the graphical user interface to prioritize the applications. In one embodiment, the prioritization includes specifying a preferred application from the applications that source device **400** can access. In another embodiment, the prioritization includes ordering from most important to least important every application that source device **400** can access. In yet another embodiment, the prioritization includes ordering from most important to least important a portion of the applications that source device **400** can access. Once the applications are prioritized, source device **400** sends an inquiry request to the ad-hoc communications network (step **429**). Peer device **450**, one of the devices in the ad-hoc communications network that is in inquiry scan mode, receives the inquiry request (step **467**) and responds by sending an inquiry response message (step **468**). In one embodiment, the inquiry response message is a Bluetooth inquiry result command modified to indicate that peer device **450** includes a middleware layer. Source device **400** receives the inquiry response message (step **430**). Source device **400** examines the inquiry response message to determine whether the inquiry response message includes an indication that peer device **450** may include the middleware layer (step **431**). If the inquiry response message does not include the indication, the process exits. If the inquiry response message includes the indication, source device **400** accesses the combined application directory (step **432**) and examines the control parameters associated with the prioritized applications (step **433**). The control parameters provide several user-defined alternatives for each application, as well as, the ability to combine alternatives into a macro. In one embodiment, the control parameter alternatives include allowing a connection between the selected application running on source device **400** and a matching application running on peer device **450** (step **434**). If the connection is not allowed, the process exits. If the connection is allowed, source device **400** conducts paging and service discovery with peer device **450** to establish a link connection (step **435** and step **469**). Following establishment of the link connection, source device **400** and peer device **450** launch the preferred application and begin communication (step **436** and step **470**).

[0059] Figure **5** is a flow diagram of an embodiment of a process that illustrates the message flow during establishment of a communication session between terminal X and terminal Y in a mobile ad-hoc communications network. In one embodiment, terminal X and terminal Y are mobile devices such as terminal **120** shown in Figure **1** and Figure **2B**. In another embodiment, terminal X is a mobile device such as terminal **120** shown in Figure **1** and Figure **2B** and terminal Y is a mobile device such as server **110** shown in Figure **1** and Figure **2A**.

[0060] As shown in Figure **5**, terminal X initiates the communication by sending an inquiry request message to the mobile ad-hoc communications network. Since terminal Y is a nearby device, terminal Y receives the inquiry request message and sends an inquiry response message to terminal X. In one embodiment, the inquiry request message is a Bluetooth inquiry command and the inquiry response message is a Bluetooth inquiry result command. In another embodiment, the inquiry request message is a Bluetooth inquiry command and the inquiry response message is a Bluetooth inquiry result command modified to indicate that the terminal sending the Bluetooth inquiry result command includes a middleware layer. In one embodiment, the middleware layer includes dedicated middleware software providing advanced application and service discovery and execution. In one embodiment, the modification to the Bluetooth inquiry result command is to the Class of Device (CoD) parameters. For example, if the terminal sending the Bluetooth inquiry result command includes the middleware layer, the terminal will set at least the "ad-hoc networking aware" bit (bit 16) to on (1). Alternatively, if the terminal sending the Bluetooth inquiry result command includes the middleware layer, the terminal will set the "ad-hoc networking aware" bit (bit 16) to on (1), and the "location info" bit (bit 17) to off (0). Alternatively, if the terminal sending the Bluetooth inquiry result command includes the middleware layer, the terminal will set the "ad-hoc networking aware" bit (bit 16) to on (1), and the "telephony capable" bit (bit 22) to on (1). Alternatively, if the terminal sending the Bluetooth inquiry result command includes the middleware layer, the terminal will set the "ad-hoc networking aware" bit (bit 16) to on (1), the "location info" bit (bit 17) to off (0), and the "telephony capable" bit (bit 22) to on (1). In yet another embodiment, the modification to the Bluetooth inquiry result command is not necessary, if a dedicated indication parameter to indicate the presence of the middleware software is introduced to the Bluetooth inquiry result command specifications.

[0061] Following the inquiry, as shown in Figure **5**, terminal X may create a connection to each nearby device indicating possible possession of the middleware layer by the inquiry response message, such as terminal Y, by sending a paging request message. If terminal Y does not indicate possible possession of the middleware layer (e.g., by setting the "ad-hoc networking aware" bit (bit 16) to off (0)), no paging request message is transmitted and the communication session is disconnected. After conducting an inquiry including an indication that terminal Y possibly includes a middleware layer, terminal X sends the paging message request, as discussed above. Terminal Y receives the paging request message and optionally sends a paging accept message to accept the connection request. In one embodiment, the paging request message is a Bluetooth create connection command and the paging accept message is a Bluetooth accept connection request command.

[0062] Following the connection to each nearby device, as shown in Figure **5**, terminal X sends a recognition request message to confirm whether a nearby device such as terminal Y definitely includes the middleware layer. Terminal Y receives the recognition request message and sends a recognition response message to terminal X. In one embodiment, the receipt of the recognition response message is confirmation that terminal Y includes the middleware layer. In another embodiment, the content of the recognition response message will indicate whether terminal Y includes the middleware layer. In one embodiment, the recognition request message and the recognition response message utilize the Bluetooth Service Discovery Protocol (SDP). If terminal Y does not include the middleware layer, the communication session may be disconnected.

[0063] Following the confirmation that a nearby device such as terminal Y includes the middleware layer, as shown in Figure **5**, terminal X and terminal Y use the middleware layer to discover and launch applications and services. In one embodiment, terminal X and terminal Y use the methods disclosed in the flow diagrams shown in Figures **3A–3B** and Figures **4A–4D** to discover and launch applications and services.

[0064] Although the disclosed embodiments describe a fully functioning system and method for launching and controlling application programs resident in wireless devices in a mobile ad-hoc communications network, the reader should understand that other equivalent embodiments exist. Since numerous modifications and variations will occur to those who review this disclosure, the system and method for launching and controlling application programs resident in wireless devices in a mobile ad-hoc communications network is not limited to the exact construction and operation illustrated and disclosed. Accordingly, this disclosure intends all suitable modifications and equivalents to fall within the scope of the claims.